Comments from Roy Thornton on TeleNova proposals dated 1986-05-30

1,2 and 5: Alias, Array Assignment and Multiple Assignment

The ideas seem very desirable and have common ground.

I do not like assigning the values with the DIM statement and would suggest another keyword be used. I personally quite like the word ALLOCATE though this may have implications from other languages. (Perhaps ASSIGN would be more favoured)

Examples

ALIAS

```
100 DIM a(100)
110 ALLOCATE a(1:) := b(1:) // The elements of b are copied to a()
120 ALLOCATE a(11:) := b(1:) // The elements of b are copied into
a() starting from element a(11)
130 ALLOCATE a(11:20) := b(1:) // The elements of b are copied
into a() from element a(11) to element a(20)
140 ALLOCATE a(11:) := b(5:8),c(6:7,3:9) // A list of elements to
be
copied using the 'right first' increment rule
```

ARRAY ASSIGNMENT

Constants

```
200 ALLOCATE a(1:100) := 1 // Much as proposed
201 ALLOCATE a() := 1
```

I feel strongly that wherever an array is used, the brackets should be specified, though I would not demand the commas to show number of dimensions.

Computed values

```
210 ALLOCATE a(x,y) := (x=y) // optional brackets
```

The value assigned has to be computed from the function or relationship given on the right of the assignment with x and y taken across the full range of subscripts. x and y would be local. There should be no confusion with a statement a(x,y) := (x=y) which assigns only one element.

The ALLOCATE statement in this example is the equivalent of

```
210 FOR j := minimum j to maximum j DO
211   FOR k := minimum k to maximum k DO
212     a(j,k):= (j=k) // optional brackets
213   NEXT k
214 NEXT j
```

but without the need to specify the bounds of the array or the loop variables.

MULTIPLE ASSIGNMENT

```
300 ALLOCATE a(),b(),c() := 1
```

3 and 4: PRINTing an array and PRINT USING

These seem sensible proposals though again I would wish to see the brackets for the array. eg PRINT a()

6 Substring (0)

Whilst it is entertaining that a$(0:0) appears to be at the end of the string it could just as easily be argued that a$(0:0) should be at the beginning! I am uneasy about the possible confusion and feel that this is a discovery that should cause a passing smile and not be a specified part of the language.

7 print separators

I agree totally with the idea of a symbol for new-line. I see a lot more problem about which symbol it should be. The ' is another possible candidate.

I do not see the reason for preventing a print list consisting only of separators. It may be possible to do without it, but it seems as valid an idea as a PRINT TAB(5,17); statement preceding an INPUT routine.

8 Compound statements

This seems reasonable though will inevitably lead to some messy program lines.

9 and 10 MAXIMUM and MINIMUM

The inclusion seems very desirable but could we not accept just MAX and MIN as acceptable keywords.

11 SIGMA

Great idea, but it is essential that one dimension of an array could be dealt with on its own, without having to recourse to an ALLOCATE type of rewrite. Similarly it should be possible to specify the range of elements to be summed.

eg I don't want to have to do

```
1000 DIM copy(100)
1010 ALLOCATE copy(j):= array(3,j)
1020 sum_squares := SIGMA( copy(),2)
```

I would prefer something like

```
1000 sum_squares := SIGMA( copy(3,j),2)
```

where a variable in any dimension implies the sum along that dimension.

1000 partial_squares := SIGMA(array(4,j,k),2) would mean sum across the full range of the dimensions indicated by the variables.

Also 1000 selected_sqares := SIGMA(array(2,1:5,2:4),2) should be an acceptable statement.

12 STR$

Interesting and no objections.

13 REF operator

I am sure there are good reasons for doing this, that others are better qualified to comment on.

14? Subarrays

In principle I agree with the idea.